**Question**:

You are given the following skeleton file and your job is to edit it according to the instructions below and in the .py file.

**File:**

[section6.py](https://psu.instructure.com/courses/2171620/files/135085894?wrap=1)[Download section6.py](https://psu.instructure.com/courses/2171620/files/135085894/download?download_frd=1)

**Task:**

You are a TA for 131, and you implemented a class ExamTracker to automate mundane tasks. Your to-do list for today has the following tasks:

1. Complete **print\_vars().**This function doesn't take any external parameter, nor returns anything. In the body of the function, it simply prints the properties/variables of the object.
2. Complete **changeTotalStudents().**This function also doesn't take any external parameter, nor returns anything. Within the body, it first checks if the value in students is less than the total length of grades. If it is less, then replace the value in students with the length of grades.
3. Complete **changeSection().**This function doesn't take any external parameter, nor returns anything. Within the body of the function, do this:
   * If the first element of classInfo is 0, assign the second element of classInfo to the variable named section
   * If the first element of classInfo is 1, assign the third element to classInfo to the  variable named section
4. Complete **addCurve().**This function takes an external parameter named points, but doesn't return anything. Within the body, it simply goes over the list of grades and adds points to each grade.
5. Complete **classStatus().**This function doesn't takes any external parameter, but returns a string explaining how your class is doing. Within the body of the function, it does the following:
   * If the mean is less than or equal to 0 or greater than 100, return "Error in code"
   * If the mean is greater than 0 and less than or equal to 40, return "Class is doing poorly"
   * If the mean is greater than 40 and less than 60, return "Class is doing somewhat better"
   * If the mean is greater than or equal to 60 and less than 100,  return "Class has a high mean"
6. Invoke all functions according to the comments in the attached file.

Code:

class ExamTracker:

  def \_\_init\_\_(self, grades, section, students, mean, classInfo):

    self.grades=grades #List that has grades of all students stored in it

    self.section=section #A string that has the section name

    self.students=students #An integer that represents total number of students

    self.mean=mean #Mean of the grades

    self.classInfo=classInfo #A tuple that would manipulate self.section

  def print\_vars(self):

    #This function doesn't take any param nor returns anything. It simply prints all of the properties of the object

    print(self.grades, self.section, self.students, self.mean, self.classInfo)

  def changeTotalStudents(self):

    #If self.students is greater than the length of self.grades, reassign self.students to the length of grades

    #This function also doesn't take any param, nor returns anything

    if self.students>len(self.grades):

      self.students=len(self.grades)

  def changeSection(self):

    #This function doesn't take a param, nor returns anything

    #It uses self.classInfo, which is a tuple with three elements e.g. (0, "Section 006", "CMPSC-131-Section 006")

    #If first element of self.classInfo is 0, change self.section to the second element of self.classInfo i.e. (0, "Section 006", "CMPSC-131-Section 006")------> self.section becomes Section 006

    #If first element is 1, then self.section gets the third element of the tuple

    if self.classInfo[0]==0:

      self.section=self.classInfo[1]

    elif self.classInfo[0]==1:

      self.section=self.classInfo[2]

  def addCurve(self, points):

    #This function takes an external parameter named points and adds that to each element of self.grades

    #This function doesn't return anything

    for i in range(len(self.grades)):

      self.grades[i]+=points

  def classStatus(self):

    #This function doesn't take anything but returns how a class is doing

    #If the mean is less than or equal to 0, return "Error in code"

    #If the mean is greater than 0 and less than 40, return "Class is doing poorly"

    #If the mean is equal to 40 and less than 60, return "Class is doing somewhat better"

    #If the mean is greater than 60 or equal to 60, return "Class has a high mean"

    if self.mean<=0:

      return "Error in code"

    elif self.mean>0 and self.mean<40:

      return "Class is doing poorly"

    elif self.mean>=40 and self.mean<60:

      return "Class is doing somewhat better"

    elif self.mean>=60:

      return "Class has a high mean"

#--------------------------------------------------------------------------------

#-----------DO NOT CHANGE THIS---------------------------------------------------

grades = [60, 44, 35, 88, 95, 11, 66]

section = "006"

students = 87

mean = 44.5

classInfo = (1, "Section 006", "CMPSC-131-Section 006")

exam = ExamTracker(grades, section, students, mean, classInfo)

#--------------------------------------------------------------------------------

#Invoke print\_vars

#Invoke changeTotalStudents

#Invoke print\_vars

#Invoke changeSection

#Invoke print\_vars

#Invoke addCurve

#Invoke print\_vars

#Invoke classStatus

#Print the returned result after invocation

exam.print\_vars()

exam.changeTotalStudents()

exam.print\_vars()

exam.changeSection()

exam.print\_vars()

exam.addCurve(10)

exam.print\_vars()

print(exam.classStatus())